MILCOM 2016

SECURE COMMUNICATIONS AT THE SPEED OF CYBER

# Scheduling and Path Planning for Computational Ferrying

Sebastian A. Zanlongo
Alexander Wilson
Leonardo Bobadilla
Tamim Sookoor

BALTIMORE, MD • NOVEMBER 1–3, 2016

# Introduction

- Units operating in complex, dynamic, adversarial environments need access to significant computational resources.
- This computation must be performed quickly and securely.

https://www.army.mil/article/124829/Apps_on_a_map_link_the_command_post__mounted_and_dismounted_Soldiers

# Motivation

- Only able to carry small devices to maintain mobility.
- These devices are expected to execute complex tasks with limited:
  - Processing power
  - Battery life
  - Time limits

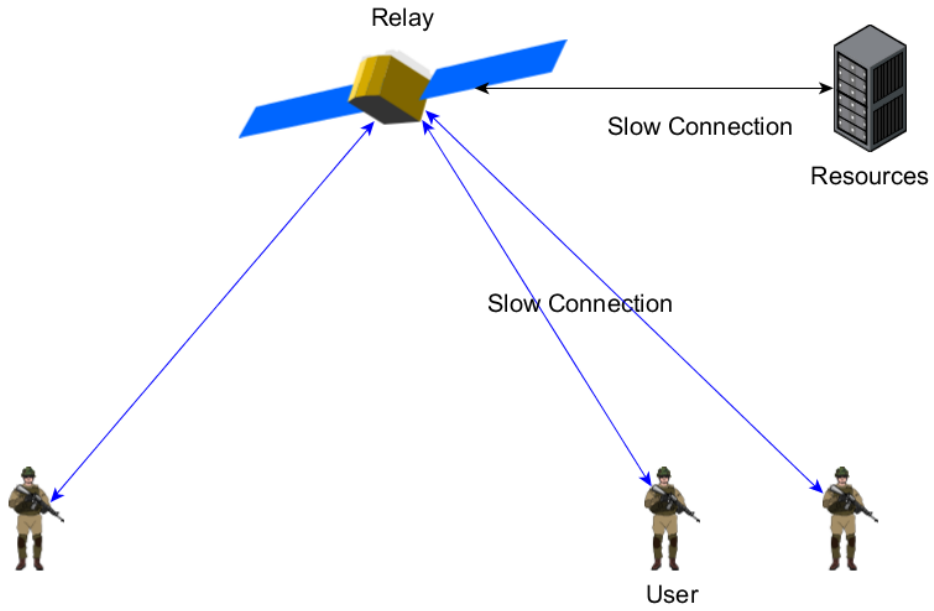
https://www.army.mil/article/124829/Apps_on_a_map_link_the_command_post__mounted_and_dismounted_Soldiers

# Problem

- How can we provide units in the field with timely access to large computational capability?

# Related Work

- Cloud computing platforms [4, 8] promise to solve all these issues given fast connections.
- Environments may not allow for high-speed communications.




[4] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload,"
[8] M. S. Gordon, D. A. Jamshidi, S. A. Mahlke, Z. M. Mao, and X. Chen. Comet: Code offload by migrating execution transparently.

# Related Work

- Traditional ferrying approaches aim to solve the problem of transporting messages from one location to another [9, 17, 22].
- The trajectories followed by MHPCs in [13] are direct lines between pickup and delivery locations.

[9] S. Guo and S. Keshav. Fair and efficient scheduling in data ferrying networks.
[13] A. Monfared, M. Ammar, E. Zegura, D. Doria, and D. Bruno. Computational ferrying: Challenges in deploying a mobile high performance computer.
[17] A. Skordylis and N. Trigoni. Delay-bounded routing in vehicular ad-hoc networks.
[22] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks.

# Solution

- Our work expands on computational ferrying [13].
- Offload computation from mobile devices onto moving cloudlets.

[13] A. Monfared, M. Ammar, E. Zegura, D. Doria, and D. Bruno, “Computational ferrying: Challenges in deploying a mobile high performance computer,”

# Solution

- Utilize Mobile High Performance Computer (MHPC) to take computation to the user.
- Provide:
  - Greater processing power
  - Large storage capability
  - Power savings

# Problem Definition

- Generate a path $\tau$ for each MHPC $A$ such that we can fulfill all tasks $T$ while reducing the number of deadlines missed.

# Physical State Space

- MHPC's $A$ move in a 2D, partially known workspace $W = \mathbb{R}^2$
- Have known and unknown obstacles:
  - $O = O_{known} \cup O_{unknown}$
  - $O \in W$
- State space $X = X^1 \times \cdots \times X^m$ representing all configurations of MHPC's $A = A^1 \times \cdots \times A^m$.

# Physical State Space

- MHPC-Obstacle collisions:
  - $X_{obs}^{j} = \{x \in X | A^{j}(x^{j}) \cap O \neq \emptyset\}$
- MHPC-MHPC collisions:
  - $X_{obs}^{jl} = \{x \in X | A^{j}(x^{j}) \cap A^{l}(x^{l}) \neq \emptyset\}$
- Complete obstacle region:
  - $X_{obs} = \left(\cup_{j=1}^{m} X_{obs}^{j}\right) \cup \left(\cup_{jl, j \neq l} X_{obs}^{jl}\right)$
- Valid configurations $X_{free} = X \backslash \mathrm{X_{obs}}$ are obstacle-free.

# MHPC's

- A Mobile High Performance Computer (MHPC) is similar to traditional message ferrying, with the addition of computation.
- MHPC's have a number of processors $p \geq 1$, visit order $vo$.
- Processors follow an execution schedule for tasks and report progress to a central controller.
- MHPC's are capable of sensing their environment and of limited path planning.
- Follow trajectory $\tilde{x}$ from $x_I$ to $x_G$ through $X_{free}$.

# Complexity

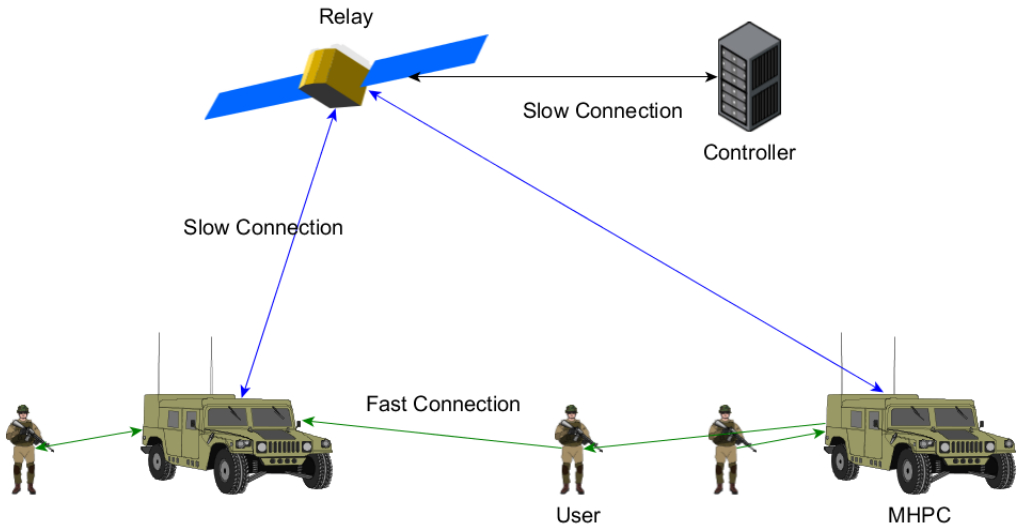
- Computational Ferrying Problem contains a known NP-hard problem; the Partition Problem, which:
  - Decides whether a set of positive integers $S$ can be split into two subsets $S^1 \subset S, S^2 \subset S$ such that the sum of numbers in each set is equal.
- We restrict our problem of computational ferrying, and set pick up and delivery locations to be the same:
  - $T^i_{LP}, T^i_{LD} = (0,0) \forall T^i \in T$

# Complexity

- All tasks are made available at $t = 0$, and deadlines are ignored.
- Assume that $A$ only consists of 2 MHPCs
  - Each has 1 processor.
  - Their locations are the same as those of the tasks.
- These restrictions remove the element of travel, arriving to the Partition Problem where the positive integers are the task lengths.

# System Architecture

- MHPCs
- Users produce and consume tasks.
- A controller responsible for:
  - Receiving user requests
  - Receiving MHPC status
  - Creating MHPC roadmaps
  - Scheduling MHPC trajectories
  - Scheduling tasks across MHPC processors



# Methods [Overview]

1. Using known static obstacles, task locations, and MHPC locations, create roadmap and paths.
2. Scheduling algorithm uses tentative paths to create MHPC ordering.
3. MHPC's carry out scheduled visit order.
4. If new tasks arrive, or MHPC's are added or removed: reschedule and create new paths.

# Methods [Timing]

- Estimate: pickup, start, finish, and delivery times of the tasks based on:
  - MHPC location,
  - Task pickup/delivery locations
  - Task job length
  - MHPC processor schedule
  - MHPC visit order
  - Estimated travel distance

# Methods [Scheduling]

1. The group of tasks is sorted by their deadlines.
2. Permutations are generated for each task's placement over all MHPC's.
3. Set each task's pickup, start, finish, and delivery times.
4. Reset the optimal solution with the visit order that meets the most requirements.
5. After every task has been placed in the best ordering over the group of MHPC's, the final solution is used to create a hybrid path.

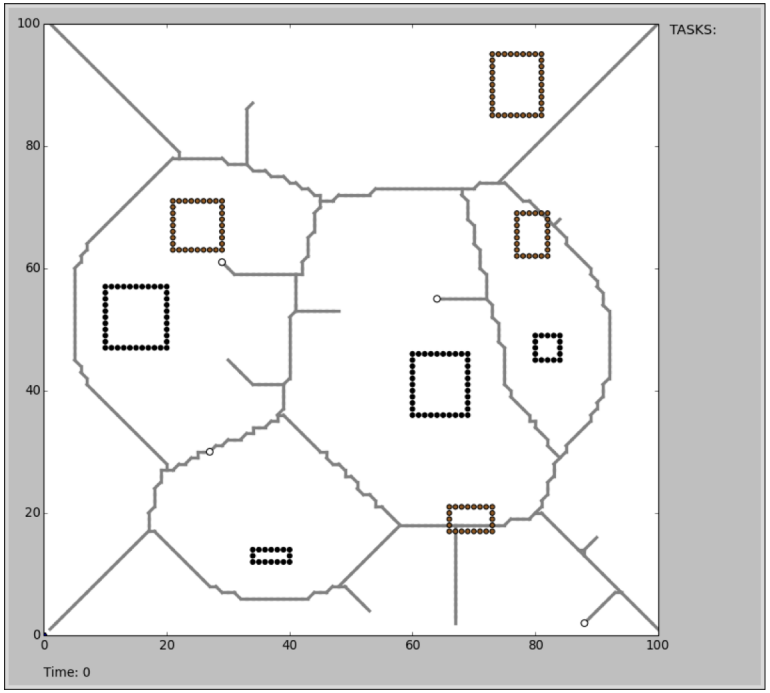
**Algorithm 1** Task Group Scheduling Algorithm

Input: $T$, $\mathcal{M}$
Output: $\mathcal{M}$ representing greedy solution

```
1: T' = Sort T by ascending deadlines
2: for T^i ∈ T' do
3:     orderings = []
4:     for M^j ∈ M do
5:         pList = placementGeneration(M^j, T^i)
6:         for p ∈ pList do
7:             timedPermInfo = taskTiming(M^j, p)
8:             orderings.append(timedPermInfo)
9:     Sort orderings by user conditions
10:    Assign tasks to MHPCs based on orderings[0]
   return M
```

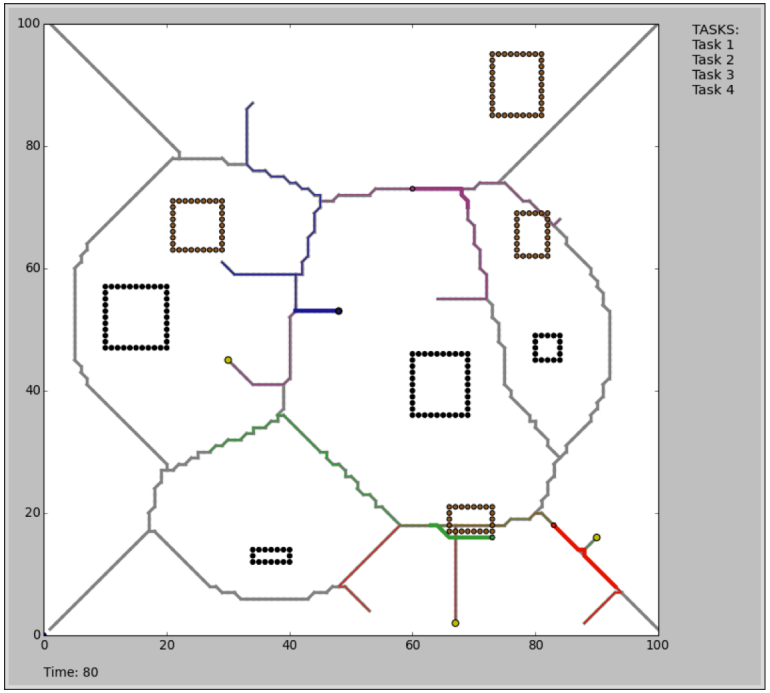
# Methods [Path Planning]

- Controller takes as input the set of known obstacles and uses a Voronoi decomposition to generate a roadmap of “safe” paths
  - Allows for efficient pathfinding later on.
  - As new points come in, they are added to roadmap.



# Methods [Path Planning]

- A tentative path is then generated along the roadmap.
  - However this does not account for unknown obstacles.
  - Use path splicing to generate a path around obstacles.



# Methods [Path Planning]

Path planning is executed for all tasks in the MHPC's queue to derive a final trajectory by chaining the separate trajectories together into a single, continuous plan.

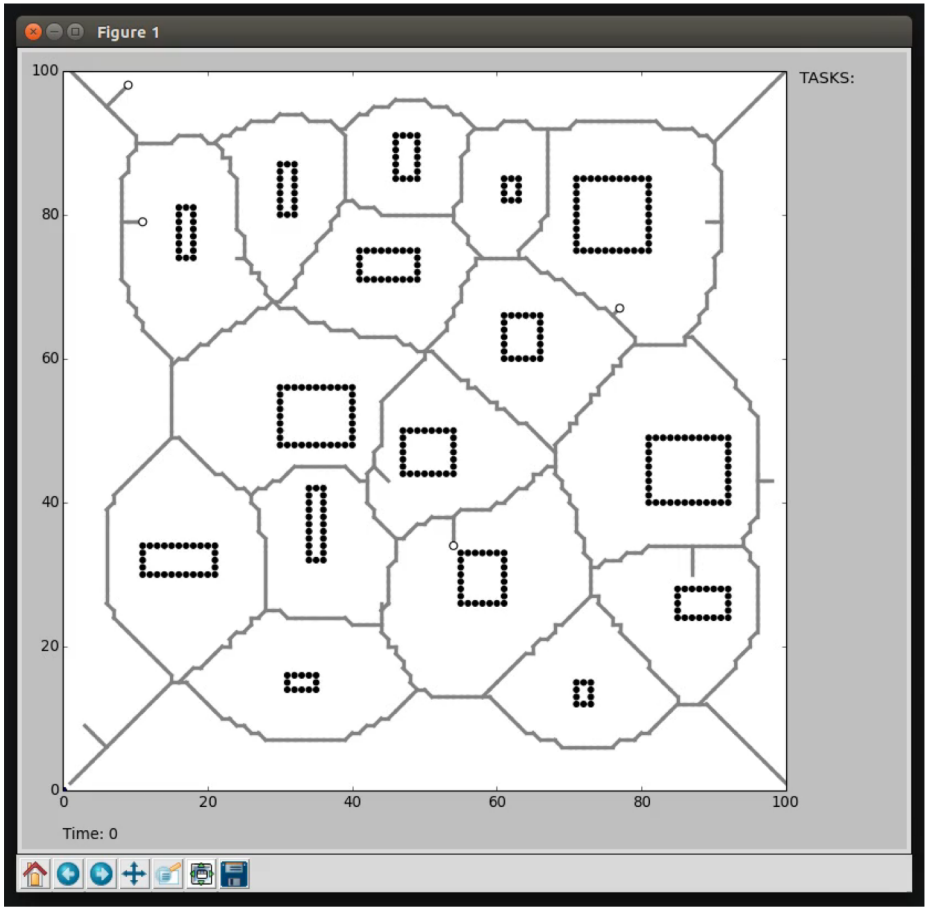
**Algorithm 2** MHPC Tentative Path

Input: $\mathcal{M}^j$
Output: Tentative path representing MHPC trajectory

```
tentativePath = []
prevLoc = M^j_x
nextLoc = Null
scheduled = []
for T^i ∈ M^j_vo do
    if T^i_PU = true then
        nextLoc = T^i_LD
    else
        nextLoc = T^i_LP
        scheduled[i] = true
    τ^σ = graphSearch(prevLoc, nextLoc)
    tentativePath.append(τ^σ)
    prevLoc = nextLoc
return tentativePath
```

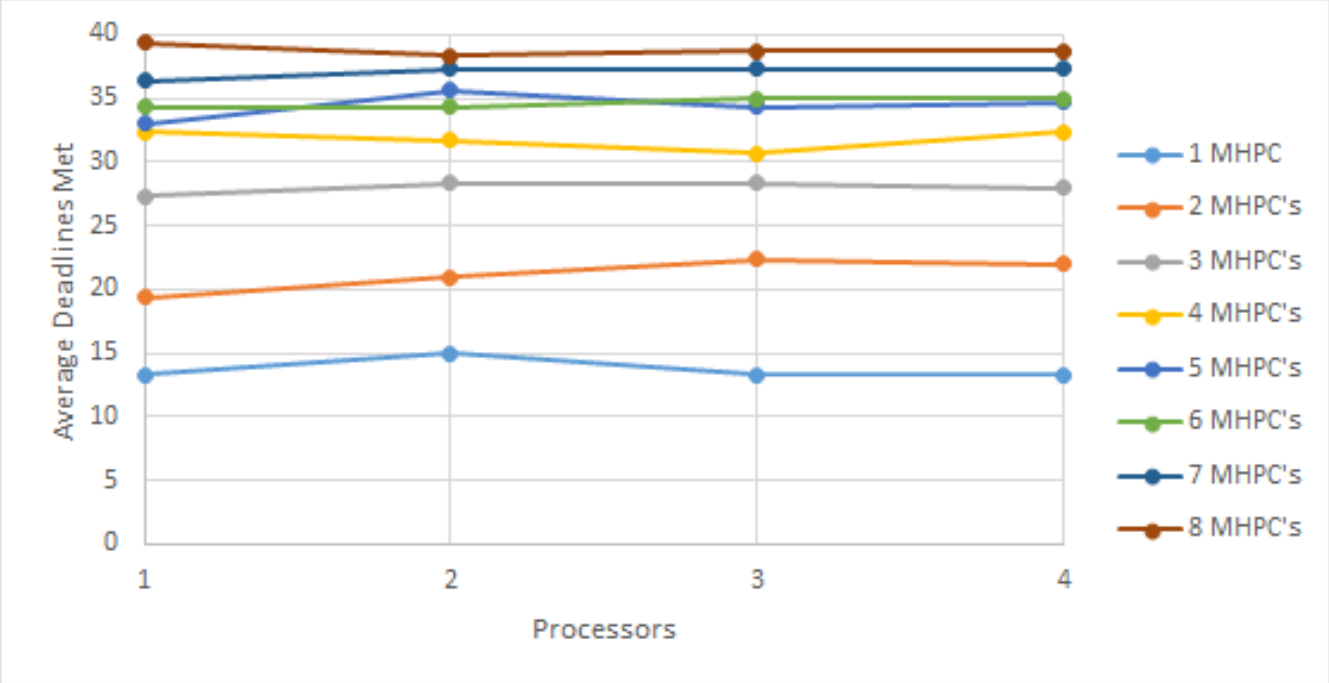
# Simulation

| Item | Combinations |
|---|---|
| MHPCs | 1-8 |
| CPU's/MHPC | 1-4 |
| Tasks | Priority 1 – 10<br>Priority 2 – 10<br>Priority 3 – 10<br>Priority 4 - 10 |
| Obstacles | 0 known, 0 unknown<br>8 known, 0 unknown<br>0 known, 8 unknown |



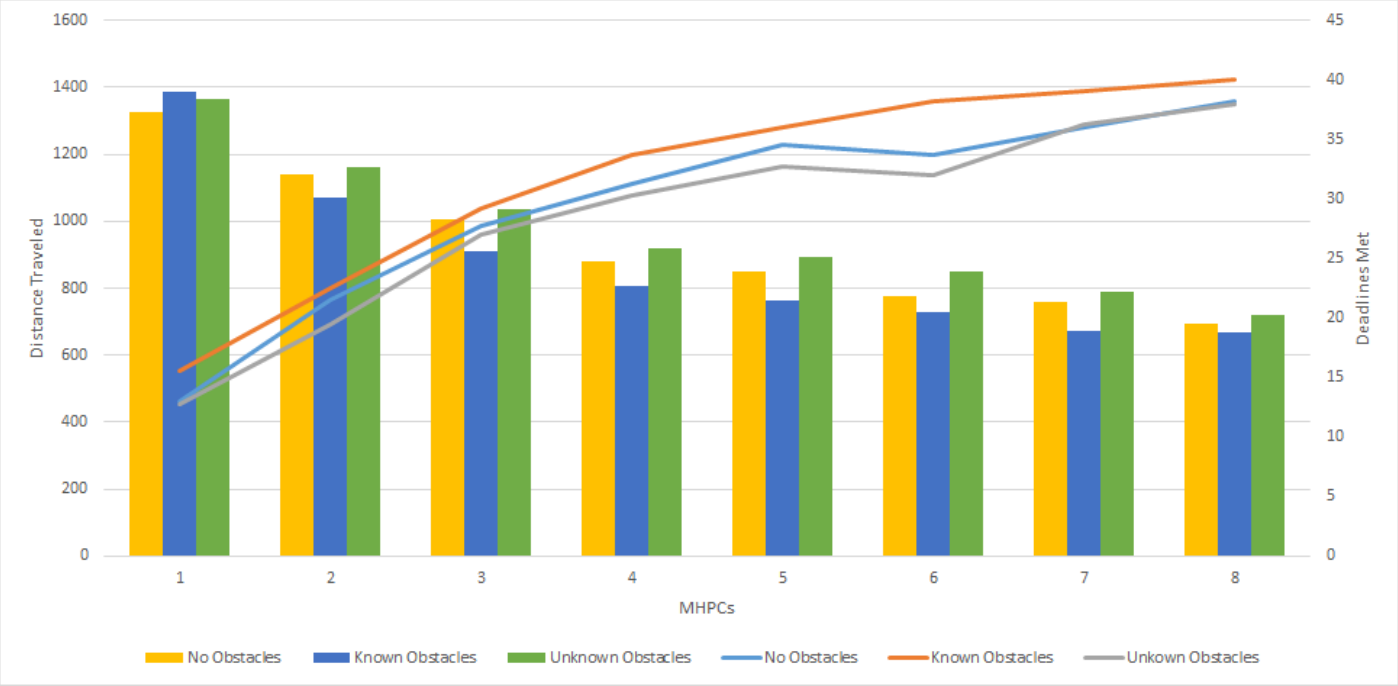
# Simulation Results

Effect of Number of MHPCs and CPUs on Task Deadlines Met



# Simulation Results

## Effects of Obstacles on MHPC Movement and Task Deadlines



# Contributions

## Improvements

- MHPC's can have different numbers of processors,
- More realistic distance measure.
- Enhanced penalty function by introducing multiple prioritized criterion.
  - Task priorities
  - MHPC specializations

## Contributions

- Task deadlines
- Prioritization by number of deadlines met.
- Environment with obstacles and pathfinding [23].
- Dynamic obstacle avoidance
- MHPC's can be added and removed.

[23] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in INFOCOM 2005. 24th annual joint conference of the IEEE computer and communications societies. Proceedings IEEE, vol. 2. IEEE, 2005, pp. 1407–1418.

# Conclusion

- There is a need for efficient offloading of large tasks to Mobile High Performance Computers.
- Computational ferrying provides a possible solution for the execution of large computational tasks in adverse environments.
- By merging the scheduling algorithm with a path-estimation and path-finding solution, we can more reliably determine an outcome where the MHPC's meet the most deadlines.

# Future Work

- Scheduling and pathfinding for MHPC's to meet and exchange information amongst themselves (P2P).
- Allow for tasks of unknown duration.
- Merging Voronoi maps and heuristics when path planning.

[22] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing. ACM, 2004, pp. 187–198.

# Acknowledgements


- G2-Inc.
- DHS Scientific Leadership Awards for Minority Serving Institutions Granting Graduate Degrees
  - Funding Opportunity Number: DHS-10-ST-062-003
- 2015 Intel/HENAAC Scholars Program
- FIU Applied Research Center (ARC) and Department of Energy Office of Environmental Management Science and Technology Workforce Development Initiative
  - DOE-FIU Cooperative Agreement DE-EM0000598

# Thank You

- Questions?
- Comments?